Sri Lanka Institute of Information Technology

Hospital Management System

MLB_04.2_10

Group Details:

	Student Registration Number	Student Name
1	IT19069432	Dissanayake D.M.I.M.
2	IT19064932	Hameed M. S.
3	IT19060736	Tilakaratne W. M. A. D. S. S.
4	IT19051208	Sakalasooriya S. A. H. A.
5	IT19051376	Anjana W.W.M.

1. Part 1

1.1. User stories

As a **Patient**
I want to **Make online appointments**
So that I can **Channel doctors easily**

As a **Patient**
I want to **View doctor availability**
So that I can **Select a proper date and time according to the doctor's availability**

As a **Doctor**
I want to **See Patients' medical history**
So that I can **Identify diseases and treat patients accurately**

As a **Doctor**
I want to **Add prescription lists**
So that I can **prescribe drugs to the patients which will be saved in their accounts**

As a **Doctor**
I want to **See my schedule**
So that I can **Make necessary changes**

As a **System Administrator**
I want to **Manage users**
So that I can **Add or remove users whenever it is necessary**

As a **System Administrator**
I want to **Edit staff schedules**
So that I can **Keep the system up to date**

As a **System Administrator**
I want to **Receive Summery Reports**
So that I can **Stay up to date about current system status**

As a **Pharmacist**
I want to **Get patients' prescription lists**
So that I can **issue drugs to the patients**

As a **Receptionist**
I want to **Make online payments**
So that I can **Pay patients' bills**

1.2. Noun/ Verb Analysis

Noun/ Noun Phrase	**Type**
Patient	***Class***
Appointment	***Class***
Doctor	***Class***
Staff	***Class***
Schedule	***Class***
Receptionist	***Class***
Pharmacist	***Class***
System Administrator	***Class***
Doctor Availability	*Out of scope*
Appointment Date	*Attribute*
Appointment Time	*Attribute*
Patient Medical History	*Redundant*
Diseases	*Out of Scope*
Treatments	***Class***
Prescription list	***Class***
Account	*Redundant*
Drugs	*Out of Scope*
Users	*Redundant*
Staff Schedule	*Redundant*
System	*Out of Scope*
Report	***Class***
System state	*Out of Scope*
Payment	***Class***
Bill	***Class***

1.3. CRC Cards

Class Name: Patient	
Responsibilities:	**Collaborations:**
Add patient	
Delete Patient	
Make appointments	Appointment
Set patient details	
Edit patient details	
Get patient details	
Check appointment details	Appointment
Edit appointment details	Appointment
Delete appointments	Appointment
Check available doctors	Schedule
View medical history	Treatment
Get payment details	Payment
Make Payment	Payment
Get Bill details	Bill
Get prescription history	Prescription list

Class Name: Appointment	
Responsibilities:	**Collaborations:**
Set Appointment details	
Get Appointment details	
Delete Appointments	
Add appointment	
Generate bill	Bill

Class Name: Staff	
Responsibilities:	**Collaborations:**
Get staff details	

Class Name: Doctor	
Responsibilities:	**Collaborations:**
Set doctor details	Staff
Get doctor details	Staff
Check schedule	Schedule
Edit schedule	Schedule
Check patients' medical history	Treatment
Check Appointments	Appointment
Add prescriptions	Prescription list

Class Name: Schedule	
Responsibilities:	**Collaborations:**
Add Schedule	
Delete Schedule	
Set Schedule details	
Get Schedule details	

Class Name: Receptionist	
Responsibilities:	**Collaborations:**
Get receptionist details	Staff
Set receptionist details	Staff
Make payment	Payment
Check bill	Bill

Class Name: Pharmacist	
Responsibilities:	**Collaborations:**
Set pharmacist details	Staff
Get pharmacist details	Staff
Check prescription details	Prescription

Class Name: System Administrator	
Responsibilities:	**Collaborations:**
Add users	Patient/ Staff
Delete user	Patient/ Staff
Set user details	Patient/Staff
Get user details	Patient/ Staff
Check reports	Report
Check schedules	Schedule
Edit Schedules	Schedule
Delete Schedules	Schedule

Class Name: Treatments	
Responsibilities:	**Collaborations:**
Add treatment Set treatment details Get treatment details Delete treatment	

Class Name: Prescription list	
Responsibilities:	**Collaborations:**
Add prescription list Set prescription details Get prescription details Delete prescription list	

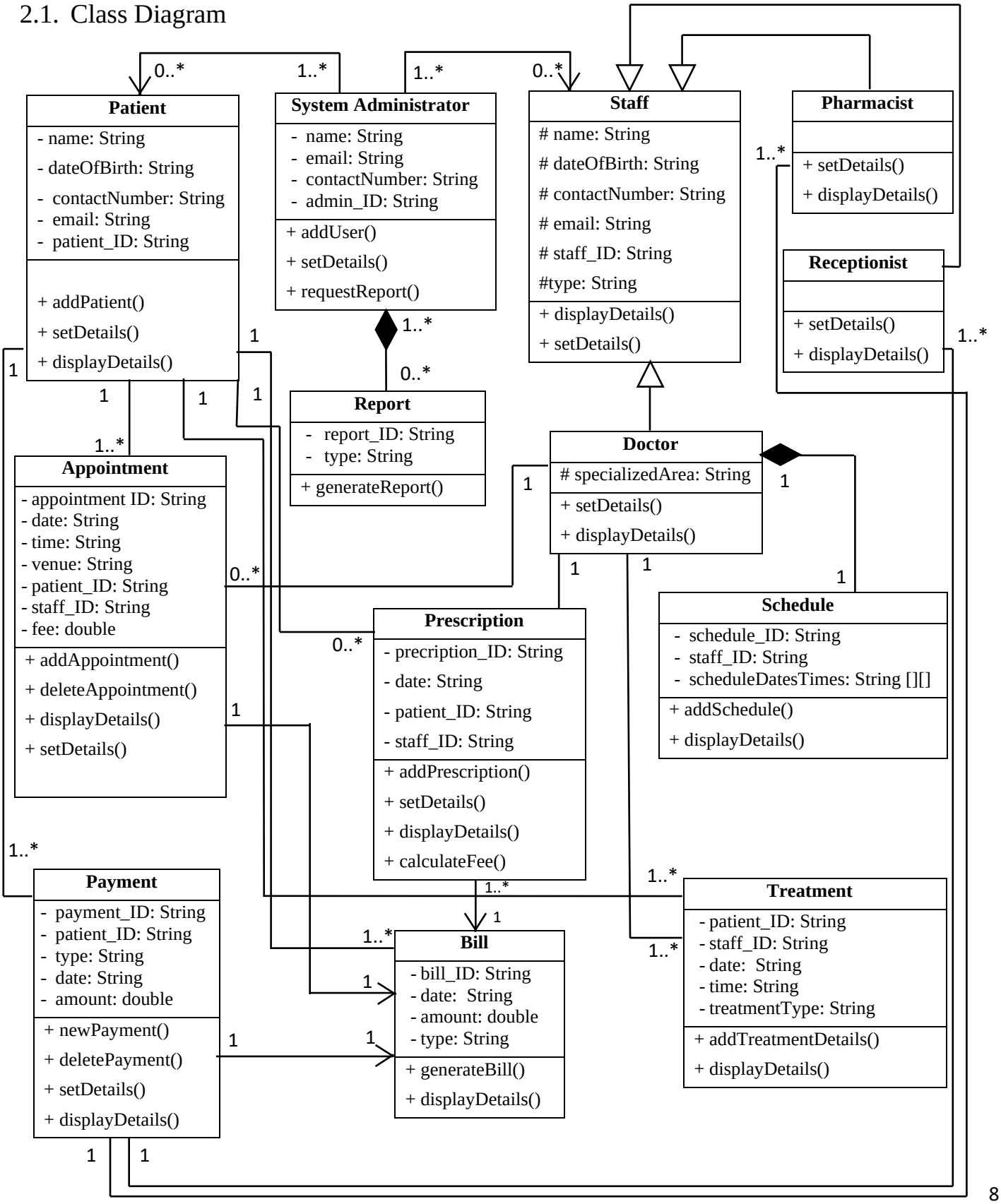
Class Name: Payment	
Responsibilities:	**Collaborations:**
Add payment Delete payment Set payment details Get payment details	

Class Name: Bill	
Responsibilities:	**Collaborations:**
Add bill Delete bill Set bill details Get bill details	

Class Name: Report	
Responsibilities:	**Collaborations:**
Generate report	
Get report details	
Check user details	Patient/ Staff
Check payment details	Payment
Check bill details	Bill
Check appointment details	Appointment

2. Part 2

2.1. Class Diagram



2.2. Code for each Class

2.2.1. Main.cpp

```
// Main.cpp : This file contains the 'main' function. Program execution begins and
ends there.

#include "SystemAdmin.h"
#include "Staff.h"
#include "Doctor.h"
#include "Appointment.h"
#include "Pharmacist.h"
#include "Receptionist.h"
#include "Treatment.h"
#include "Payment.h"

#include <iostream>
using namespace std;

int main()
{
    //create new user objects of admin, doctor,receptionist,pharmacist and patient
    cout << "Creating New User Objects" << endl;
    cout << "-------------------------" << endl;

    SystemAdmin* admin = new SystemAdmin("AID001");
    Staff* stf1 = new Doctor("Doctor");
    Staff* stf2 = new Receptionist("Receptionist");
    Staff* stf3 = new Pharmacist("Pharmacist");
    Patient* p1 = new Patient("PID001");

    //admin tasks
    cout << "System Administrator Tasks are running" << endl;
    cout << "--------------------------------------" << endl;

    //request reports
    admin->requestReport();

    admin->addUser(stf1);
    admin->addUser(stf2);
    admin->addUser(stf3);
    admin->addUser(p1);

    //create appointment Object(s)
    Appointment* app = new Appointment();
    app->addAppointment("APP001",p1, stf1,"2020-02-02","10:00 AM", "A311", 500.00);
    app->displayDetails();
    //generateBill
    app->showBill();

    //creating new treatment
    Treatment* tr = new Treatment();
    tr->addTreatmentDetails(p1,stf1, "2020-02-02","10:20AM","Heart Diagnosis");
    tr->displayDetails();
}
```

2.2.2. Appointment.h

```
#pragma once

#ifndef _APPOINTMENT
#define _APPOINTMENT

#include <string>
//Association
#include "Doctor.h"
#include "Patient.h"
//Uni-directional association
#include "Bill.h"

using namespace std; //for string variables

//forward declaration
class Patient;
class Doctor;
class Bill;

class Appointment
{
private:
    Bill* bill;
    Patient* pat;
    Staff* stf;
    string appointmentID, date, time, venue;
    double fee = 0;

public:
    Appointment();
    ~Appointment();
    void addAppointment(string appID, Patient* pat, Staff* stf, string date, string
time, string venue, double fee);
    void displayDetails();
    void showBill();
};

#endif
```

2.2.3. Appointment.cpp

```
#include "Appointment.h"
#include <string>
#include<iostream>

using namespace std;


Appointment::Appointment() {
    this->appointmentID = "Not set";
    this->date = "Not set";
    this->time = "Not set";
    this->venue = "Not set";
}

Appointment::~Appointment() {
    cout << "Deleting Appointment " << appointmentID << endl;
}

void Appointment::addAppointment(string appID, Patient* pat, Staff* stf, string
date, string time, string venue, double fee) {
    this->appointmentID = appID;
    this->date = date;
    this->time = time;
    this->venue = venue;
    this->pat = pat;
    this->stf = stf;
    this->fee = fee;
    //bill->generateBill(pat, "BILL001", "2020-02-02", "Appointment
Fee",appointmentID, fee);
}

void Appointment::displayDetails() {
    cout << endl << "Appointment Details of : " << pat->getPatientID() << endl;
    cout << "Doctor: " << stf->getStaffID() << endl;
    cout << "Date: " << date << endl;
    cout << "Time: " << time << endl;
    cout << "Venue: " << venue << endl;
    cout << "Appointment Fee: " << fee << endl << endl;
}

void Appointment::showBill() {
    bill->displayDetails();
}
```

2.2.4. Bill.h

```
#pragma once

#ifndef _BILL
#define _BILL

#include <string>
//association
#include "Patient.h"

using namespace std; //for string variables

//forward declaration
class Patient;

class Bill
{
private:
    Patient* pat;
    string billID, date, type, aID;
    double amount;

public:
    Bill();
    ~Bill();
    void generateBill(Patient* pat, string billID, string date, string type, string
appointmentID, double amount);
    void displayDetails();
};

#endif
```

2.2.5. Bill.cpp

```
#include "Bill.h"
#include <string>
#include<iostream>

using namespace std;

Bill::Bill() {
    this->billID = "Not set";
    this->date = "Not set";
    this->amount = 0.00;
    this->type = "Not set";
}

Bill::~Bill() {
    cout << "Deleting Bill " << billID << endl;
}

void Bill::generateBill(Patient* pat, string billID, string date, string type,
string appointmentID, double amount) {
    this->aID = appointmentID;
    this->pat = pat;
    this->date = date;
    this->type = type;
    this->billID = billID;
    this->amount = amount;
}


void Bill::displayDetails() {
    cout << endl << "Bill Details of : " << pat->getPatientID() << endl;
    cout << "Bill ID: " << billID << endl;
    cout << "Appointment ID: " << aID << endl;
    cout << "Date: " << date << endl;
    cout << "Type: " << type << endl;
    cout << "Amount: " << amount << endl << endl;
}
```

2.2.6. Doctor.h

```
#pragma once

#ifndef _DOCTOR
#define _DOCTOR

#include <string>
#include<iostream>
//Inheritance
#include "Staff.h"
//Association
#include "Treatment.h"
#include "Appointment.h"
#include "Prescription.h"
//including composition relationship(s)
#include "Schedule.h"

using namespace std; //for string variables

//forward declaration
class Prescription;
class Appointment;
class Treatment;

class Doctor : public Staff
{
protected:
    string specialization;
    Schedule* sch;
    Treatment* tmt;
    Appointment* apmt;
    Prescription* pres;
public:
    Doctor(string type);
    ~Doctor();
    void addUser(string specialization, string staffID, string name, string dob,
string email, string cn);
    void setDetails(string specialization, string name, string dateOfBirth, string
contactNumber, string email, string staffID);
    void displayDetails();
};

#endif
```

2.2.7. Doctor.cpp

```
#include "Doctor.h"
#include <string>
#include<iostream>

using namespace std;

Doctor::Doctor(string type) : Staff(type) {
    this->type = type;
}

Doctor::~Doctor(){
    cout << "Deleting Doctor " << staffID << endl;
}

void Doctor::setDetails(string specialization, string name, string dateOfBirth,
string contactNumber, string email, string staffID) {
    this->specialization = specialization;
    this->name = name;
    this->dateOfBirth = dateOfBirth;
    this->contactNumber = contactNumber;
    this->email = email;
    this->staffID = staffID;
}


void Doctor::displayDetails() {
    cout << endl << "Details of " << staffID << endl;
    cout << "Name: " << name << endl;
    cout << "Specialization: " << specialization << endl;
    cout << "Date of Birth: " << dateOfBirth << endl;
    cout << "Email: " << email << endl;
    cout << "Contact Number: " << contactNumber << endl << endl;
}

void Doctor::addUser(string specialization, string staffID, string name, string dob,
string email, string cn) {
    this->staffID = staffID;
    this->name = name;
    this->dateOfBirth = dob;
    this->email = email;
    this->contactNumber = cn;
    cout << "Please enter the specialized area of the doctor: ";
    cin >> this->specialization;

    cout << "New Doctor: " << staffID << " added successfully!" << endl;
}
```

2.2.8. Patient.h

```
#pragma once

#ifndef _PATIENT
#define _PATIENT

#include <string>
//association
#include "Prescription.h"
#include "Appointment.h"
#include "Treatment.h"
#include "Payment.h"

//uni-directional association
#include "Bill.h"

using namespace std; //for string variables

//forward declaration
class Payment;
class Appointment;
class Treatment;
class Prescription;

class Patient
{
private:
    string name, dateOfBirth, contactNumber, email, patientID;
    Payment* pmt;
    Treatment* tmt;
    Appointment* apmt;
    Prescription* pres;
public:
    Patient(string patientID);
    ~Patient();
    string getPatientID();
    void addPatient(string patientID, string name, string dob, string email, string
cn);
    void displayDetails(void);
    void setDetails(string name, string dateOfBirth, string contactNumber, string
email);
};

#endif
```

2.2.9. Patient.cpp

```
#include "Patient.h"
#include <string>
#include<iostream>

using namespace std;

Patient::Patient(string patientID) {
    this->patientID = patientID;
    this->name = "Not set";
    this->dateOfBirth = "Not set";
    this->email = "Not set";
    this->contactNumber = "Not set";
}

Patient::~Patient() {
    cout << "Deleting Patient " << patientID << endl;
}

void Patient::displayDetails(void) {
    cout << endl << "Details of Patient: " << patientID << endl;
    cout << "Name: " << name << endl;
    cout << "Date of Birth: " << dateOfBirth << endl;
    cout << "Email: " << email << endl;
    cout << "Contact Number: " << contactNumber << endl << endl;
}

void Patient::setDetails(string name, string dateOfBirth, string contactNumber,
string email) {
    this->name = name;
    this->dateOfBirth = dateOfBirth;
    this->contactNumber = contactNumber;
    this->email = email;
}

void Patient::addPatient(string patientID, string name, string dob, string email,
string cn) {
    this->patientID = patientID;
    this->name = name;
    this->dateOfBirth = dob;
    this->email = email;
    this->contactNumber = cn;
    cout << "New Patient: " << patientID << " added successfully!" << endl;
}

string Patient::getPatientID() {
    return patientID;
}
```

2.2.10. Payment.h

```
#pragma once

#ifndef _PAYMENT
#define _PAYMENT

//association
#include "Patient.h"
#include "Pharmacist.h"
#include "Receptionist.h"
//uni-directional association
#include "Bill.h"

using namespace std; //for string variables

//forward declaration
class Patient;
class Pharmacist;
class Receptionist;

class Payment
{
private:
    string paymentID, staffID, patientID, type, date;
    double amount = 0;
    Patient* patient;
    Staff* staff;

public:
    Payment();
    ~Payment();
    void newPayment(Patient* pat, Staff* stf, string date, string type, double
amount);
    void displayDetails();
};

#endif
```

2.2.11. Payment.cpp

```
#include "Payment.h"
#include <string>
#include<iostream>

using namespace std;

Payment::Payment() {
    this->patientID = "Not set";
    this->staffID = "Not set";
    this->paymentID = "Not set";
    this->date = "Not set";
    this->type = "Not set";
    this->amount = 0;
}

Payment::~Payment() {
    cout << "Deleting payment: " << paymentID << endl << endl;
}

void Payment::newPayment(Patient* pat, Staff* stf,string date, string type, double
amount) {
    this->patient = pat;
    this->staff = stf;
    this->date = date;
    this->type = type;
    this->amount = amount;
}


void Payment::displayDetails() {
    cout << endl << "Payment Details of: " << patient->getPatientID() << endl;
    cout << "Patient ID: " << patient->getPatientID() << endl;
    cout << "Staff ID: " << staff->getStaffID() << endl;
    cout << "Date: " << date << " "  << endl;
    cout << "Amount: " << amount << " LKR" << endl << endl;
}
```

2.2.12. Pharmacist.h

```
#pragma once

#ifndef _PHARMACIST
#define _PHARMACIST

#include <string>
#include<iostream>
//Inheritance
#include "Staff.h"
//association
#include "Payment.h"

using namespace std; //for string variables

//forward declaration
class Payment;

class Pharmacist : public Staff
{
protected:
    Payment* pmt = NULL;
public:
    Pharmacist(string type) : Staff(type) {};
    ~Pharmacist();
    void setDetails(string name, string dateOfBirth, string contactNumber, string
email, string staffID);
    void displayDetails();
};

#endif
```

2.2.13. Pharmacist.cpp

```
#include "Pharmacist.h"
#include <string>
#include<iostream>

using namespace std;

Pharmacist::~Pharmacist() {

}

void Pharmacist::setDetails(string name, string dateOfBirth, string contactNumber,
string email, string staffID) {
    this->name = name;
    this->dateOfBirth = dateOfBirth;
    this->contactNumber = contactNumber;
    this->email = email;
    this->staffID = staffID;
}

void Pharmacist::displayDetails() {
    cout << endl << "Details of " << staffID << endl;
    cout << "Name: " << name << endl;
    cout << "Date of Birth: " << dateOfBirth << endl;
    cout << "Email: " << email << endl;
    cout << "Contact Number: " << contactNumber << endl << endl;
}
```

2.2.14. Prescription.h

```
#pragma once

#ifndef _PRESCRIPTION
#define _PRESCRIPTION

#include <string>

//association
#include "Doctor.h"
#include "Patient.h"
//uni-directional association
#include "Bill.h"

using namespace std; //for string variables

//forward declaration
class Doctor;
class Patient;

class Prescription
{
private:
    Staff* stf;
    Patient* pat;
    string prescriptionID;

public:
    Prescription();
    ~Prescription();
    void addPrescription(Patient* pat, Staff* stf, string prescriptionID);
    void displayDetails();
};

#endif
```

2.2.15. Prescription.cpp

```
#include "Prescription.h"
#include <string>
#include<iostream>

using namespace std;

Prescription::Prescription() {
    this->prescriptionID = "Not Set";
}

Prescription::~Prescription() {
    cout << "Deleting Prescription " << prescriptionID << endl;
}

void Prescription::addPrescription(Patient* pat, Staff* stf, string prescriptionID)
{
    this->prescriptionID = prescriptionID;
    this->pat = pat;
    this->stf = stf;
}


void Prescription::displayDetails() {
    cout << endl << "Prescription Details of: " << pat->getPatientID() << endl;
    cout << "Prescription ID: " << prescriptionID << endl;
    cout << "Doctor ID: " << stf->getStaffID() << endl << endl;
}
```

2.2.16. Receptionist.h

```
#pragma once

#ifndef _RECEPTIONIST
#define _RECEPTIONIST

#include <string>
#include<iostream>
//Inheritance
#include "Staff.h"
//association
#include "Payment.h"

using namespace std; //for string variables

//forward declaration
class Payment;

class Receptionist : public Staff
{
protected:
    Payment* pmt = NULL;
public:
    Receptionist(string type) : Staff(type) {};
    ~Receptionist();
    void setDetails(string name, string dateOfBirth, string contactNumber, string
email, string staffID);
    void displayDetails();
};

#endif
```

2.2.17. Receptionist.cpp

```
#include "Receptionist.h"
#include <string>
#include<iostream>

using namespace std;

Receptionist::~Receptionist() {
    cout << "Deleting Receptionist " << staffID << endl;
}

void Receptionist::setDetails(string name, string dateOfBirth, string contactNumber,
string email, string staffID) {
    this->name = name;
    this->dateOfBirth = dateOfBirth;
    this->contactNumber = contactNumber;
    this->email = email;
    this->staffID = staffID;
}

void Receptionist::displayDetails() {
    cout << endl << "Details of Receptionist: " << staffID << endl;
    cout << "Name: " << name << endl;
    cout << "Date of Birth: " << dateOfBirth << endl;
    cout << "Email: " << email << endl;
    cout << "Contact Number: " << contactNumber << endl << endl;
}
```

2.2.18. Report.h

```
#pragma once

#ifndef _REPORT
#define _REPORT

#include <string>

using namespace std; //for string variables

class Report
{
private:
    string reportID;
public:
    Report();
    ~Report();
    void generateReport();
};

#endif
```

2.2.19. Report.cpp

```
#include "Report.h"
#include <string>
#include<iostream>

using namespace std;

Report::Report() {
    this->reportID = "Not set";
}

Report::~Report() {
    cout << "Deleting Report " << endl;
}

void Report::generateReport() {
    string rType;
    cout << "Please enter report type: " << endl;
    cout << "\tU - User details" << endl;
    cout << "\tS - Salary details" << endl;
    cout << "\tC - Schedule details" << endl << ": ";
    cin >> rType;

    //check requested report type and display
    cout << endl << "------------------------------------------------------------" <<
endl;
    if (rType == "U" || rType == "u") {
           cout << "User Details Report" << endl;
           cout << "This is a report generated and displayed by Report class" <<
endl;
    }
    else if (rType == "S"|| rType == "s") {
           cout << "Salary Details Report" << endl;
           cout << "This is a report generated and displayed by Report class" <<
endl;
    }
    else if (rType == "C" || rType == "c") {
           cout << "Staff Schedule Details Report" << endl;
           cout << "This is a report generated and displayed by Report class" <<
endl;
    }
    else {
           cout << "Error Generating Report" << endl;
           cout << "The requested report cannot be generated. Please double check
the type." << endl;
    }

    cout << "------------------------------------------------------------" << endl <<
endl;
}
```

2.2.20. Schedule.h

```
#pragma once

#ifndef _SCHEDULE
#define _SCHEDULE

#include <string>

using namespace std; //for string variables

class Schedule
{
private:
    string scheduleID;
    string scheduleDatesTimes[7][1];
public:
    Schedule();
    ~Schedule();
    void addSchedule();
    void displayDetails();

};

#endif
```

2.2.21. Schedule.cpp

```
#include "Schedule.h"
#include <string>
#include<iostream>

using namespace std;

Schedule::Schedule() {
    //Set default values
    this->scheduleID = "Not set";
    for (int i = 0; i < 7; i++) {
           for (int j = 0; j < 1; j++) {
                  scheduleDatesTimes[i][0] = "Date Not Set";
                  scheduleDatesTimes[i][1] = "Time Not Set";
           }
    }
}

Schedule::~Schedule() {
    cout << "Schedule ["<< scheduleID <<"] deleted";
}

void Schedule::addSchedule() {
    this->scheduleID = "SCH001";
    //default schedule
    scheduleDatesTimes[0][0] = "Monday";
    scheduleDatesTimes[1][0] = "Tuesday";
    scheduleDatesTimes[2][0] = "Wednesday";
    scheduleDatesTimes[3][0] = "Thursday";
    scheduleDatesTimes[4][0] = "Friday";
    scheduleDatesTimes[5][0] = "Saturday";
    scheduleDatesTimes[6][0] = "Sunday";
    scheduleDatesTimes[0][1] = "7:30-9:30 AM";
    scheduleDatesTimes[1][1] = "7:30-9:30 AM";
    scheduleDatesTimes[2][1] = "10:30 AM-12:30 PM";
    scheduleDatesTimes[3][1] = "1:30-4:30 PM";
    scheduleDatesTimes[4][1] = "7:30-9:30 AM";
    scheduleDatesTimes[5][1] = "Leave";
    scheduleDatesTimes[6][1] = "Leave";
}

void Schedule::displayDetails() {

    cout << "Schedule: " << scheduleID << endl;
    for (int i = 0; i < 7; i++) {
           for (int j = 0; j < 1; j++) {
                  cout << scheduleDatesTimes[i][j] <<"\t";
           }
           cout << endl;
    }
    cout << endl << endl;
}
```

2.2.22. Staff.h

```
#pragma once

#ifndef _STAFF
#define _STAFF

#include <string>

using namespace std; //for string variables

class Staff
{
protected:
    string name, dateOfBirth,contactNumber, email, staffID, type;

public:
    Staff(string type);
    ~Staff();
    string getStaffID();
    string getType();
    virtual void addUser(string staffID, string name, string dob, string email,
string cn);
    virtual void displayDetails(void);
    virtual void setDetails(string name, string dateOfBirth, string contactNumber,
string email, string staffID);
};

#endif
```

2.2.23. Staff.cpp

```
#include "Staff.h"
#include <string>
#include<iostream>

using namespace std;

//Default constructor requires staff type when creating objects
Staff::Staff(string type) {
    this->type = type;
    this->name = "Not set";
    this->dateOfBirth = "Not set";
    this->email = "Not set";
    this->contactNumber = "Not set";
    this->staffID = "Not set";
}

Staff::~Staff() {
    cout << "Deleting Staff Member " << staffID << endl;
}

void Staff::displayDetails(void) {
    cout << endl << "Details of Staff member: " << staffID << endl;
    cout << "Name: " << name << endl;
    cout << "Date of Birth: " << dateOfBirth << endl;
    cout << "Email: " << email << endl;
    cout << "Contact Number: " << contactNumber << endl << endl;
}

void Staff::setDetails( string name, string dateOfBirth, string contactNumber,
string email, string staffID) {
    this->name = name;
    this->dateOfBirth = dateOfBirth;
    this->contactNumber = contactNumber;
    this->email = email;
    this->staffID = staffID;
}

void Staff::addUser(string staffID, string name, string dob, string email, string
cn) {
    this->staffID = staffID;
    this->name = name;
    this->dateOfBirth = dob;
    this->email = email;
    this->contactNumber = cn;
    cout << "New Staff Member: " << staffID << " added successfully!" << endl;
}

string Staff::getStaffID() {
    return staffID;
}

string Staff::getType() {
    return type;
}
```

2.2.24. SystemAdmin.h

```
#pragma once

#ifndef _SYSTEMADMIN
#define _SYSTEMADMIN

#include <string>
//including composition relationship(s)
#include "Report.h"

//including uni-directional association relationships
#include "Patient.h"
#include "Staff.h"

using namespace std; //for string variables

class SystemAdmin
{
private:
    string name, email, contactNumber, adminID;
    Report* rep;

public:
    SystemAdmin(string adminID);
    ~SystemAdmin();
    void setDetails(string name, string email, string contactNumber);
    void addUser(Patient* pat);
    void addUser(Staff* stf);
    void requestReport();
};

#endif
```

2.2.25. SystemAdmin.cpp

```
#include "SystemAdmin.h"
#include <string>
#include<iostream>

using namespace std;

SystemAdmin::SystemAdmin(string adminID) {
    this->adminID = adminID;
    this->name = "Not set";
    this->email = "Not set";
    this->contactNumber = "Not set";
}

SystemAdmin::~SystemAdmin() {
    cout << "Deleting System Administrator Object" << endl << endl;
}

void SystemAdmin::setDetails(string name, string email, string contactNumber) {
    this->name = name;
    this->email = email;
    this->contactNumber = contactNumber;
}
//add patient
void SystemAdmin::addUser(Patient* pat) {
    string name, patientID, email, cn, dob;
    cout << "Please enter Patient Details:" << endl;
    cout << "\tEnter PatientID: ";
    cin >> patientID;
    cout << "\tEnter Name: ";
    cin >> name;
    cout << "\tEnter Email: ";
    cin >> email;
    cout << "\tEnter Date of Birth: ";
    cin >> cn;
    cout << "\tEnter Contact Number: ";
    cin >> dob;
    cout << endl << endl;

    pat->addPatient(patientID, name, dob, email, cn);
    pat->displayDetails();
}
```

```
//addStaffMember
void SystemAdmin::addUser(Staff* stf) {
    string staffID, name, email, cn, dob;
    cout << "Please enter " << stf->getType() << " Details:" << endl;
    cout << "\tEnter "<< stf->getType() << "ID: ";
    cin >> staffID;
    cout << "\tEnter Name: ";
    cin >> name;
    cout << "\tEnter Email: ";
    cin >> email;
    cout << "\tEnter Date of Birth: ";
    cin >> cn;
    cout << "\tEnter Contact Number: ";
    cin >> dob;
    cout << endl << endl;

    stf->addUser(staffID, name, dob, email,cn);
    stf->displayDetails();
}

void SystemAdmin::requestReport() {
    //request new report
    rep->generateReport();
}
```

2.2.26. Teatment.h

```
#pragma once

#ifndef _TREATMENT
#define _TREATMENT

#include <string>
//association
#include "Patient.h"
#include "Doctor.h"

using namespace std; //for string variables

//forward declaration
class Doctor;
class Patient;

class Treatment
{
private:
    string patientID, staffID, date, time, treatmentType;
    Patient* pat;
    Doctor* doc;
public:
    Treatment();
    ~Treatment();
    void addTreatmentDetails(Patient* patient ,Staff* stf, string date, string time,
string type);
    void displayDetails();

};

#endif
```

2.2.27. Treatment.cpp

```
#include "Treatment.h"
#include <string>
#include<iostream>

using namespace std;

Treatment::Treatment() {
    this->patientID = "Not set";
    this->staffID = "Not set";
    this->date = "Not set";
    this->time = "Not set";
    this->treatmentType = "Not set";
}

Treatment::~Treatment() {
    cout << "Deleting Treatment " << patientID << endl;
}

void Treatment::addTreatmentDetails(Patient* patient, Staff* stf, string date,
string time, string type) {
    //this->patientID = pat-> getPatientID();
    this->staffID = stf->getStaffID();
    this->date = date;
    this->time = time;
    this->treatmentType = type;
}

void Treatment::displayDetails() {
    cout << endl << "Treatment Details of: " << patientID << endl;
    cout << "Patient ID: " << patientID << endl;
    cout << "Doctor ID: " << staffID << endl;
    cout << "Date and Time: " << date << " " << time << endl;
    cout << "Treatment Details: " << treatmentType << endl << endl;
}
```